# **Continuous Integration** (and Deployment)
# **CI/CD**

**The principles and practices**

# What is CI?

- **CI is a** software development practice **where members of a team** integrate their work frequently **– at least daily - leading to multiple integrations per day. Each integration is verified by** an automated build **(including tests) to detect integration errors as quickly as possible.** *Martin Fowler*

- **In software engineering, CI implements** continuous **processes** of applying quality control **- small pieces of effort, applied frequently. CI aims to** improve the quality **of software, and to** reduce the time **taken to deliver it, by replacing the** traditional practice **of applying quality control after completing all development.**

# Why do CI?

- **Building software can be a *risky* business.**

- **Risk 1: Fixing bugs late is** costly**.**
  - **CI identifies them as early as possibly => Less reworking.**

- **Risk 2: Lack of** project visibility.
  - **EX.: Code coverage; Test failer/success rate.**

# Why do CI?

- **Risk 3: Lack of** team cohesion**.**
  - **“Your changes to module X are incompatible with mine. How do we merge now?”**
  - **“When did we decided to upgrade to version 2.0 of library X?”**
  - **“I thought you fixed that 2 months ago!”**

- **Risk 4:** Poor quality **code base.**
  - **EX.: “Everybody knows === is safer than == for object comparison (JavaScript)”**

# Why do CI?

- **Risk 5: Lack of** deployable **software.**
  - **“It works on my machine!.”**
  - **“I need a new build to play around with now.”**
  - **“The boss (or customer) is coming, we need to demo progress asap.”.**

**=> Use** Continuous Integration **to reduce these risks.**

# Why CI? Better, Faster, Cheper

- Better**:**

  **Build better quality software …… that is tested early and often ……and the code adheres to best practices & coding standards**

- Faster:
  - **Regression testing (Agile), not just at the end (Waterfall).**
  - **No 'integration points'.**
  - **System Builds become a non event**
- Cheaper
  - **Identify defects earlier.**
  - **Fix when least costly.**

# Best practices of CI.

- **Single Source Repository.**
- **Automate the Build and Test processes.**
- **Everyone Commits Every Day.**
- **Keep the Build Fast.**
- **Everyone can see what's happening.**
- **Automate Deployment (Optional).**

# What constitutes a **build**?

1. **Compilation (Transpiling).**
   - **bundling for web apps.**
   - **Multiple target platforms.**
2. **Test exacution.**
3. **Database integration.**
   - **DB creation and test data generation**
4. **Code inspection (Static code analysis).**
5. **Automated deployment (Use staging server).**
6. **Report generation.**
   - **e.g. test execution metrics.**

# When and how to build?

- **When:**
  - **At every check-in (not scheduled times).**

- **How?**
  - **Use a build** script**. Do not depend on an IDE.**
  - **Use a dedicated** CI server**, not cron.**
  - **Should require no developer effort**
  - **Provide immediate feedback**

# Build metrics

- **Identify key metrics and track them visually.**
- **Act on them immediately.**

- **Many metrics options, including:**
  - **Successful Build Rate.**
  - **Build Repair Rate.**
  - **Total Numbers of Static Tool Errors.**
    - **Ex. Linter.**
  - **Code coverage.**
    - **Unit Testing.**
    - **Functional Testing.**

# The 7 step program

1. **Commit Early, Commit Often.**
2. **Never Commit Broken Code.**
3. **Fix build failures immediately.**
4. **Fail Fast.**
5. **Act on metrics.**
6. **Build in every target environment.**
7. **Create artifacts from every build.**

Travis CI

**The tools**

# Travis CI

- **An Open Source**
- **Distributed**
- **Build System**
- **For the Open Source community**
- *travis-ci.org* **(free) Vs** *travis-ci.com* **(enterprise)**

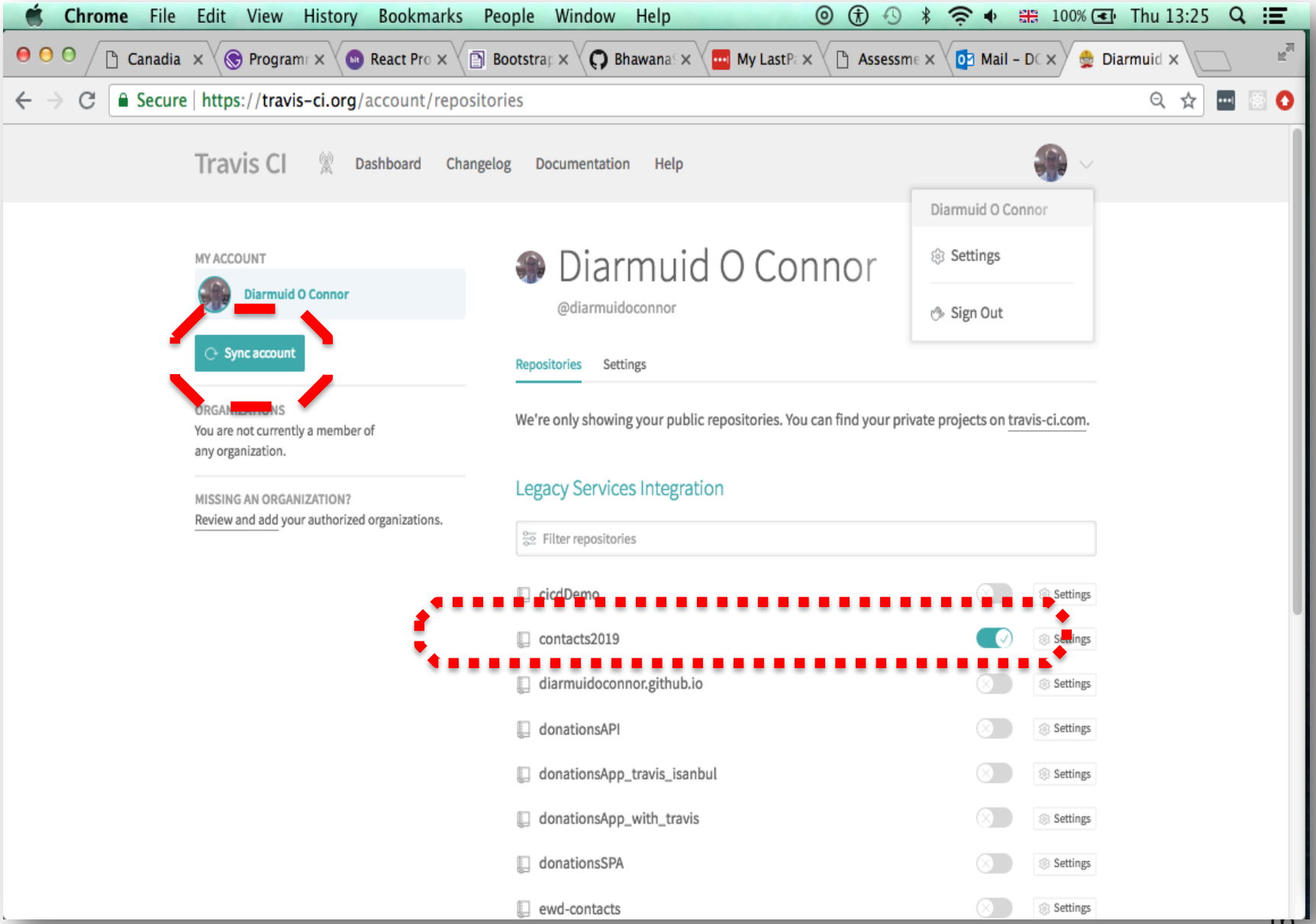
# …Distributed …

- ***. . . . allows community to contribute build capacities by connecting a VM that runs a build agent somewhere on their underused servers . . . .***

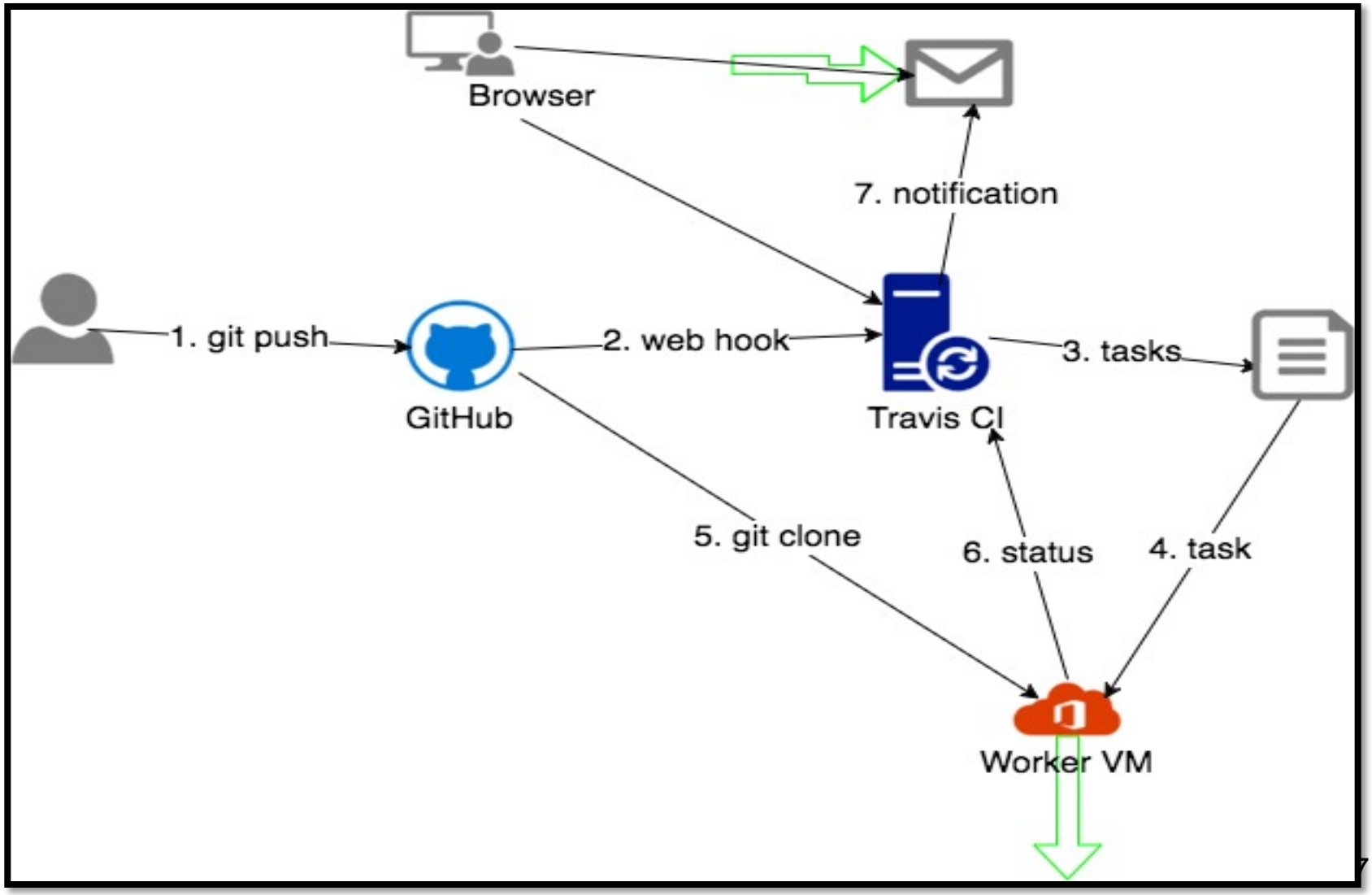
- **Travis-CI – a central** web application that **runs on Heroku.**
  - **where as Travis** workers **(Virtual Machines that run the builds) are contributed by the community.**

# Getting started

1. Sign u**p with Travis CI using your GitHub credentials.**
2. **Add the** Travis config file **to your project repository**
3. Enable **the repositories that you want Travis to perform a build when a push operation occurs**


Chrome File Edit View History Bookmarks People Window Help
Thu 13:25
Secure https://travis-ci.org/account/repositories
Travis CI
Dashboard
Changelog
Documentation
Help
Diarmuid O Connor
Settings
Sign Out
MY ACCOUNT
Diarmuid O Connor
Sync account
You are not currently a member of any organization.
MISSING AN ORGANIZATION?
Review and add your authorized organizations.
Diarmuid O Connor
@diarmuidoconnor
Repositories
Settings
We're only showing your public repositories. You can find your private projects on travis-ci.com.
Legacy Services Integration
Filter repositories
cicdDemo
contacts2019
diarmuidoconnor.github.io
donationsAPI
donationsApp_travis_isanbul
donationsApp_with_travis
donationsSPA
ewd-contacts
Settings

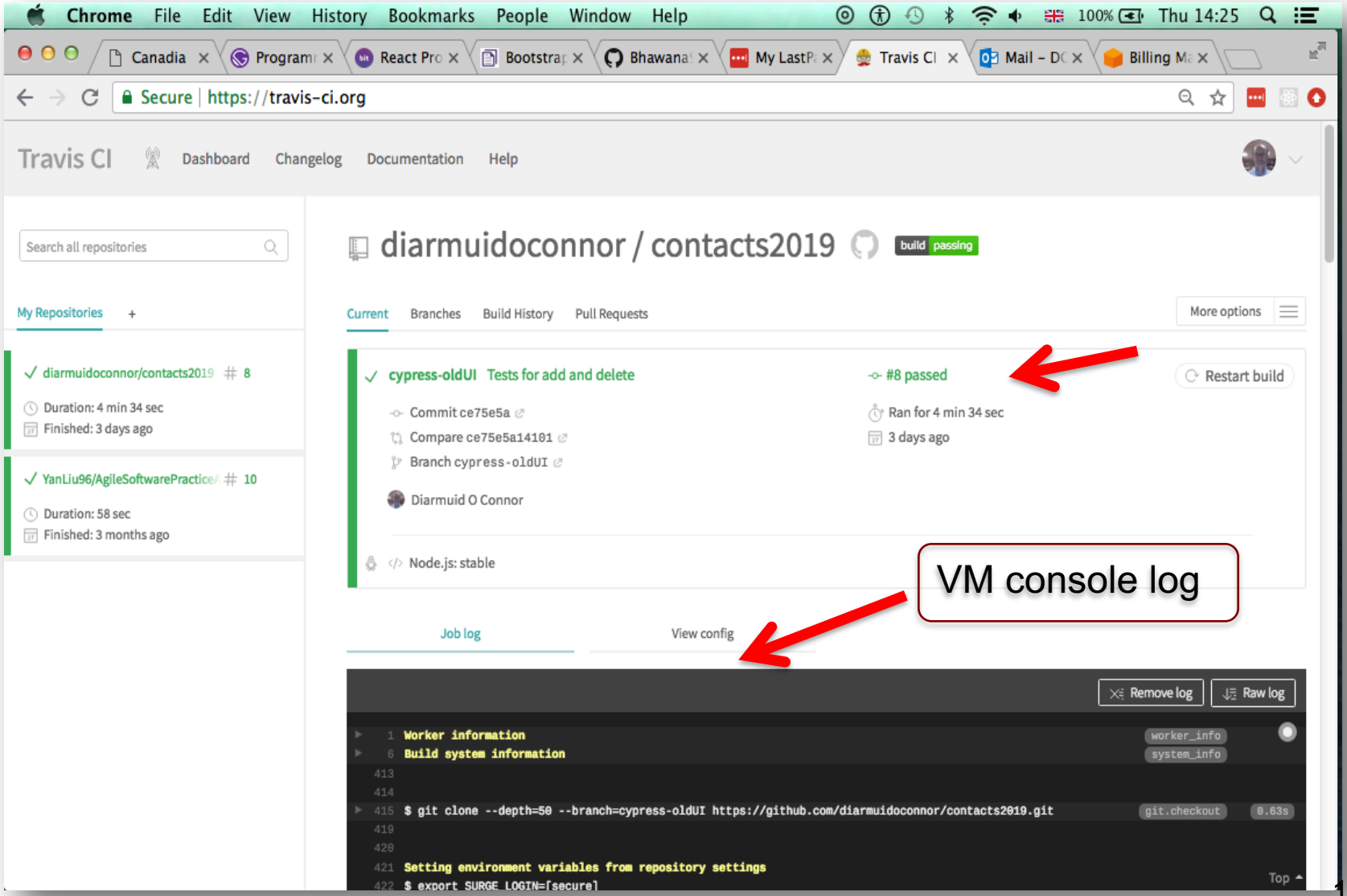
# …Build System …

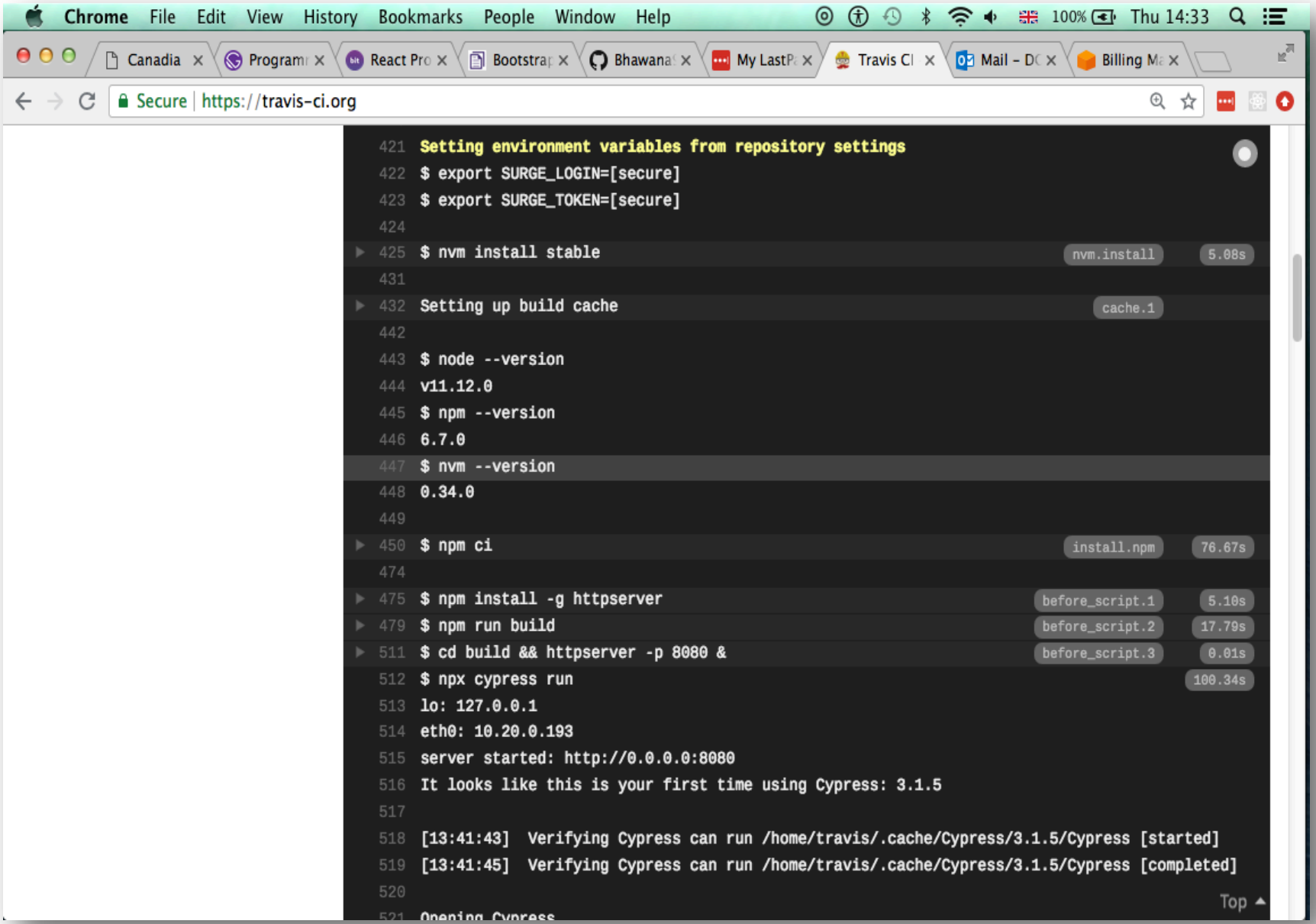


# Building with Travis

1. **In local repo, declare build requirements in special config file,** .travis.yml.
2. **git push.**
3. **Travis CI identifies available worker VM.**
4. **Sends ‘task list’ to worker.**
5. **Worker clones repository and performs tasks.**
6. **Periodically updates Travis CI of status.**

# Worker VM updates Travis CI




Chrome File Edit View History Bookmarks People Window Help
100% Thu 14:33
Canadia
Programr
React Pro
Bootstrap
Bhawana
My LastP
Travis CI
Mail - D
Billing Ma
Secure https://travis-ci.org
421 Setting environment variables from repository settings
422 $ export SURGE_LOGIN=[secure]
423 $ export SURGE_TOKEN=[secure]
424
425 $ nvm install stable nvm.install 5.08s
431
432 Setting up build cache cache.1
442
443 $ node --version
444 v11.12.0
445 $ npm --version
446 6.7.0
447 $ nvm --version
448 0.34.0
449
450 $ npm ci install.npm 76.67s
474
475 $ npm install -g httpserver before_script.1 5.10s
479 $ npm run build before_script.2 17.79s
511 $ cd build && httpserver -p 8080 &
512 $ npx cypress run 100.34s
513 lo: 127.0.0.1
514 eth0: 10.20.0.193
515 server started: http://0.0.0.0:8080
516 It looks like this is your first time using Cypress: 3.1.5
517
518 [13:41:43] Verifying Cypress can run /home/travis/.cache/Cypress/3.1.5/Cypress [started]
519 [13:41:45] Verifying Cypress can run /home/travis/.cache/Cypress/3.1.5/Cypress [completed]
520
Top
521 Opening Cypress

# Build Matrix

- **Defined by your .travis.yml**
- **Configure...**
  - **Language version**
  - **Environment Variables**
- **Script to run.**

```
# .travis.yml excerpt
language: node_js
node_js:
-'10.5'
-'11.7'
env:
-  DB='databaseA'
- DB='databaseB'
script
-npm run build
-  . . . . . .
```

# Other build features

- **before / after scripts.**
  - **e.g. (re)create test database.**

- **Specify branches to build (white/blacklist)**

```
# blacklist
branches:
  except:
    - legacy
    - experimental
```

# Languages supported

- **Ruby**
- **Node.js**
- **Scala**
- **Clojure**
- **Python**
- **……..**

# Databases supported

- **SQLite**
- **MySQL**
- **PostgreSQL**
- **MongoDB**
- **……..**

- **No extra setup required**

# Services

- **Memcached**
- **Redis**
- **Riak**
- **CouchDB**
- **Selenium**
- **ImageMagick**
- **. . . . . .**